An Algorithm for Finding Small Roots of Multivariate Polynomials over the Integers

Domingo Gómez, Jaime Gutierrez, Álvar Ibeas

Faculty of Sciences,
University of Cantabria,
Santander E–39071, Spain

Abstract. In this paper we present a new algorithm for finding small roots of multivariate polynomials over the integers based on lattice reduction techniques. Our simpler heuristic method is inspired in algorithms for predicting pseudorandom numbers, and it can be considered as another variant of Coppersmith's method for finding small solutions of integer bivariate polynomials. We also apply the method to the problem of factoring an integer when we know the high-order bits of one of the factors.

1 Introduction

In 1996, Coppersmith [9–11] introduced two rigorous lattice-based methods for finding small roots of polynomials: one for univariate modular and another one for bivariate integer polynomial equations. One of the main results is:

Theorem 1 (Theorem 2-[9]). *Let $p(\varepsilon_1, \varepsilon_2)$ be an irreducible polynomial in two variables over $\mathbb{ZZ}$, of maximum degree δ in each variable separately. Let Δ_1, Δ_2 be bounds on the desired solutions x_0, y_0. Define $p^*(\varepsilon_1, \varepsilon_2) = p(\varepsilon_1\Delta_1, \varepsilon_2\Delta_2)$ and let W be the absolute value of the largest coefficient of $p^*(\varepsilon_1, \varepsilon_2)$. If*

$$\Delta_1\Delta_2 \leq W^{2/(3\delta)-\varepsilon}2^{-14\delta/3},$$

then in polynomial time in $(\log W, \delta, 1/\varepsilon)$ we can find all integer pairs (x_0, y_0) with $p(x_0, y_0) = 0$ bounded by $|x_0| \leq \Delta_1, |y_0| \leq \Delta_2$.

The complicated proof of this important result is based on *lattice basis reduction*, with the so called LLL-technique (see [22]).

Lattice reduction techniques seem inherently linear. The general idea of this technique is to translate our non linear problem to finding a short vector in a lattice built from the nonlinear equation. Then, the so-called Shortest Vector Problem and Closest Vector Problem in lattices play a major role (see Section 2).

In recent years, these techniques have been used repeatedly for the cryptanalytic attack of various cryptosystems. Coppersmith's algorithm has many applications in cryptology: cryptanalysis of RSA with small public exponent when

some part of the message is known, polynomial time factorization of $N = PQ$ with high bits known and polynomial time factorization of $N = P^rQ$ for large r; several papers have been published on different applications of those results in cryptology, see for instance [20, 7, 12, 17, 14, 21]. The paper [9] also proposed heuristic multivariate extensions for both approaches. The goal in this kind of method is to maximize the bounds up to which roots of the polynomials can be computed in polynomial time. The recent paper [18] also presents a new strategy to solve this problem.

In this paper we present a method to compute small roots of a system of multivariate polynomial equations with integer coefficients simpler than other known algorithms for bivariate polynomials. Our heuristic algorithm is based on the same idea used for predicting non-linear pseudorandom numbers, see [4–6, 13]. Despite we are not able to provide bounds to which common roots of the polynomials can be computed, we have implemented in C++ our approach showing that it works relatively well in practice.

We also apply the method to the problem of factoring an integer when we know the high-order bits of one of the factors. In contrast to other known solutions of this problem, our approach:

- it is asymptotically less efficient,
- it does not require to compute any resultant of integer bivariate polynomials, but it requires to exclude a very small set of primes Q,
- it requires lower size lattice coefficients and smaller lattice dimension,
- it is easy to implement.

The remainder of the paper is structured as follows. We start with a short outline of some basic facts about lattices in Section 2. In Section 3 we give an overview of our heuristic algorithm, we also include some complexity and effectiveness of the proposed strategy. In Section 4 we present the results of numerical tests produced in the C++ implementation using the NTL (Number Theory Library) [25] . Section 5 discusses the problem of factoring an integer when we know the high-order bits of one of the factors.

Acknowledgment. This work is partially supported by Spanish Ministry of Science grant MTM2004-07086.

2 Background on Lattices

Here we collect several well-known facts about lattices which form the background to our algorithms.

We review several related results and definitions on lattices which can be found in [8, 15]. For more details and more recent references, we also recommend consulting [1, 16, 23, 24].

Let $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^r$. The set

$$\mathcal{L} = \{c_1\boldsymbol{b}_1 + \ldots + c_s\boldsymbol{b}_s, \quad c_1, \ldots, c_s \in \mathbb{Z}\}$$

is called an *s-dimensional lattice* with *basis* $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$. If $s = r$, the lattice $\mathcal{L}$ is of *full rank*.

To each lattice $\mathcal{L}$ one can naturally associate its *volume*

$$\mathrm{vol}(\mathcal{L}) = \left(\det\left(\langle \boldsymbol{b}_i, \boldsymbol{b}_j\rangle\right)_{i,j=1}^{s}\right)^{1/2},$$

where $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes the inner product. This definition does not depend on the choice of the basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$.

For a vector $\boldsymbol{u}$, let $\|\boldsymbol{u}\|$ denote its *Euclidean norm*. The famous Minkowski theorem gives the upper bound:

$$\min\{\|\boldsymbol{z}\| : \ \boldsymbol{z} \in \mathcal{L} \setminus \{\boldsymbol{0}\}\} \le s^{1/2}\mathrm{vol}(\mathcal{L})^{1/s} \tag{1}$$

on the shortest nonzero vector in any s-dimensional lattice $\mathcal{L}$ in terms of its volume.

The Minkowski bound (1) motivates a natural question, named *Shortest Vector Problem (SVP)*: how to find the shortest nonzero vector in a lattice. Unfortunately, there are several indications that this problem is **NP**-hard when the dimension grows. This study has suggested several definitions of a *reduced* basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ for a lattice, trying to obtain a shortest vector by the first basis element $\boldsymbol{b}_1$. The celebrated *LLL algorithm* of Lenstra, Lenstra and Lovász [22] provides a concept of *reduced* basis and a desirable solution in practice.

Another related problem is the *Closest Vector Problem (CVP)*: Given a lattice $\mathcal{L} \subseteq \mathbb{R}^r$ and a shift vector $\boldsymbol{t} \in \mathbb{R}^r$, the goal consists on finding a vector in the set $\boldsymbol{t} + \mathcal{L}$ with minimum norm. It is well-known that the CVP is **NP**-hard when the dimension grows. An approximate polynomial time solution is presented in [2].

However, both computational problems SVP and CVP are known to be solvable in deterministic polynomial time (polynomial in the bit-size of a basis of $\mathcal{L}$) provided that the dimension of $\mathcal{L}$ is fixed (see [19], for example).

3 The algorithm

In this section we present an sketch of an algorithm for computing small zeroes of a system of integer polynomial equations.

Let $f_1, \ldots, f_m$ be polynomials in the variables $x_1, \ldots, x_n$ with integer coefficients. Also all the linear monomials and one monomial of degree 2 appear in the first polynomial (this is possible thanks to Theorem 1). Suppose that the associated polynomial system of equations has an unknown common zero $(\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{Z}^n$ such that each component ε_i is bounded by some known integer bound $\Delta_i \in \mathbb{Z}$, that is, $|\varepsilon_i| \le \Delta_i$, $i = 1, \ldots n$ and

$$\begin{aligned} f_1(\varepsilon_1, \ldots, \varepsilon_n) &= 0, \\ f_2(\varepsilon_1, \ldots, \varepsilon_n) &= 0, \\ &\vdots \\ f_m(\varepsilon_1, \ldots, \varepsilon_n) &= 0. \end{aligned} \tag{2}$$

3.1 On changes of variables

In order to apply our method is necessary that all the linear monomials appear in the input polynomials. This can be done using a change of variables:

$$\begin{array}{rl} A : \mathbb{Z}[x_1, \ldots, x_n] & \to \mathbb{Z}[x_1, \ldots, x_n] \\ x_i & \to x_i + \gamma_i. \end{array}$$

The coefficient of the monomial term x_i of $A(f)$ is:

$$\frac{\partial f}{\partial x_i}(\gamma_1, \ldots, \gamma_n).$$

The roots of a polynomial and its image are related by this function:

$$\begin{array}{rccc} A' : & \mathbb{Z}^n & \to & \mathbb{Z}^n \\ & (\varepsilon_1, \ldots, \varepsilon_n) & \to & (\varepsilon_1 - \gamma_1, \ldots, \varepsilon_n - \gamma_n). \end{array}$$

If the components of a root are bounded, i. e. $|\varepsilon_i| \leq \Delta_i$ then $|\varepsilon_i - \gamma_i| \leq \Delta_i + |\gamma_i|$. Briefly, we need that

$$\frac{\partial f}{\partial x_i}(\gamma_1, \ldots, \gamma_n) \neq 0$$

and that all $|\gamma_i|$ are small. In the next theorem we will show how to find this point and give an upperbound on its component. We do not include the straightforward proof of the following result.

Proposition 1. *Given $f_1, \ldots, f_m \in \mathbb{Z}[x_1, \ldots, x_n]$, non-zero polynomials of total degree less than h. It is possible to find in polynomial time a point $(\gamma_1, \ldots, \gamma_n)$ such as*

$$|\gamma_i| \leq hm$$

and

$$f_i(\gamma_1, \ldots, \gamma_n) \neq 0$$

for all $i = 1, \ldots, m$.

Our main task is to design an efficient algorithm for computing the common zero $(\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{Z}^n$.

3.2 Plan of the algorithm

Basically, the algorithm is divided into several linearization steps.

First iteration: We construct a certain lattice $\mathcal{L}$, see (5) below. It depends on the Equation (2). We also show that a certain vector $\boldsymbol{E}$ directly related to missing information about $(\varepsilon_1, \ldots, \varepsilon_n)$ is a very short vector in the set $\boldsymbol{t} + \mathcal{L}$, where $\boldsymbol{t}$ depends on the bounds Δ_i and the coefficients of the polynomials f_i. A short vector $\boldsymbol{F}$ in $\boldsymbol{t} + \mathcal{L}$ is found; see [19, 2] for appropriate algorithms. If $\boldsymbol{F} = \boldsymbol{E}$ then the unknowns are discovered in this linearization step.

Second and more iterations: If $\boldsymbol{E} \neq \boldsymbol{F}$, then we express $\boldsymbol{E} - \boldsymbol{F}$ as a linear combination of a reduced basis of the lattice $\mathcal{L}$ with small unknown coefficients (see (6) and (7) below). Then, we apply the previous technique to the lattice associated to equations (8) with bounds given in equation (7).

3.3 Sketch of the algorithm

First iteration. We construct the lattice $\mathcal{L}$ which depends on the above polynomial system of equations. Let

$$\mathcal{M} = \{M_i(x_1, \ldots, x_n),\ i = 1, \ldots, t\}$$

be the set of all non-unit monomials of the polynomials $f_i(x_1, \ldots,\ x_n)$. So, we have that

$$f_i(x_1, \ldots, x_n) = a_{i1}M_1(x_1, \ldots, x_n) + \cdots + a_{it}M_t(x_1, \ldots, x_n) + f_i(0, \ldots, 0),$$

where $a_{ij} \in \mathbb{Z}$, $i = 1, \ldots, m$ and $j = 1, \ldots, t$. We write $M_i(x_1, \ldots,\ x_n) = y_i$ and consider the following linear system of equations with m equations in t variables y_i:

$$\begin{array}{r} a_{11}y_1 + \cdots + a_{1t}y_t = -f_1(0, \ldots, 0), \\ a_{21}y_1 + \cdots + a_{2t}y_t = -f_2(0, \ldots, 0), \\ \vdots \\ a_{m1}y_1 + \cdots + a_{mt}y_t = -f_m(0, \ldots, 0). \end{array} \tag{3}$$

We transform these equations in the following way, so that they are satisfied by a vector with balanced components and whose knowledge leads to a problem solution: let

$$\mu_i := M_i(\Delta_1, \ldots, \Delta_n),\ \Delta = \mathrm{lcm}(\mu_i, i = 1, \ldots, t).$$

From Equation (3) we see that the vector

$$\boldsymbol{E} = \left(\frac{\Delta M_1(\varepsilon_1, \ldots, \varepsilon_n)}{\mu_1}, \cdots, \frac{\Delta M_t(\varepsilon_1, \ldots, \varepsilon_n)}{\mu_t} \right)$$

is a solution of the following linear system of congruences:

$$\begin{array}{r} a_{11}\mu_1 Y_1 + \cdots + a_{1t}\mu_t Y_t = -f_1(0, \ldots, 0)\Delta, \\ a_{21}\mu_1 Y_1 + \cdots + a_{2t}\mu_t Y_t = -f_2(0, \ldots, 0)\Delta, \\ \vdots \\ a_{m1}M_1(\Delta_1, \ldots, \Delta_n)Y_1 + \cdots + a_{mt}\mu_t Y_t = -f_m(0, \ldots, 0)\Delta, \\ Y_1 \equiv 0 \mod \Delta/\mu_1, \\ \vdots \\ Y_t \equiv 0 \mod \Delta/\mu_t. \end{array} \tag{4}$$

Moreover, $\boldsymbol{E}$ is a relatively short vector. It is not difficult to prove that the Euclidean norm $\|\boldsymbol{E}\|$ of $\boldsymbol{E}$ satisfies (note that the number of monomials t is already fixed)

$$\|\boldsymbol{E}\| = O(\Delta).$$

Let $\mathcal{L}$ be the lattice consisting of integer solutions $\boldsymbol{y} = (y_1, \ldots, y_t) \in \mathbb{Z}^t$ of the system of congruences:

$$\begin{aligned} a_{11}\mu_1 Y_1 + \cdots + a_{1t}\mu_t Y_t &= 0, \\ a_{21}\mu_1 Y_1 + \cdots + a_{2t}\mu_t Y_t &= 0, \\ &\vdots \\ a_{m1}\mu_1 Y_1 + \cdots + a_{mt}\mu_t Y_t &= 0, \\ Y_1 &\equiv 0 \mod \Delta/\mu_1, \\ &\vdots \\ Y_t &\equiv 0 \mod \Delta/\mu_t. \end{aligned} \tag{5}$$

We compute any particular solution $\boldsymbol{t}$ of the linear system of congruences (4) using classical linear algebra. Now, we apply an algorithm solving the CVP for the shift vector $\boldsymbol{t}$ and the lattice $\mathcal{L}$ to obtain a vector

$$\boldsymbol{F} = \left(\frac{\Delta F_1}{\mu_1}, \cdots, \frac{\Delta F_t}{\mu_t} \right)$$

satisfying equations (4) and

$$\|\boldsymbol{F}\| = O(\Delta).$$

Note that we can compute $\boldsymbol{F}$ in polynomial time from the information we are given. We might hope that $\boldsymbol{E}$ and $\boldsymbol{F}$ are the same. We can assume without loss of generality that $x_1, x_2, \ldots, x_n \in \mathcal{M}$, otherwise we can apply a linear transformation to reach that requirement. Then, by assuming $\boldsymbol{E} = \boldsymbol{F}$ we compute ε_i and check if $(\varepsilon_1, \ldots, \varepsilon_n)$ is a solution of equations (2).

Second iteration. We compute in polynomial time a reduced basis $\{\boldsymbol{U^1}, \ldots, \boldsymbol{U^r}\}$ of the lattice $\mathcal{L}$ as explained in [22]. In fact, this computation is usually performed to solve the CVP in the previous step.

$$\boldsymbol{U^i} = \left(\frac{\Delta U_1^i}{\mu_1}, \cdots, \frac{\Delta U_t^i}{\mu_t} \right).$$

Since $\boldsymbol{E} - \boldsymbol{F} \in \mathcal{L}$ there exist integers α_i, $i = 1, \ldots, r$ such that

$$\boldsymbol{E} = \boldsymbol{F} + \sum_{i=1}^{r} \alpha_i \boldsymbol{U^i}, \tag{6}$$

and satisfying

$$|\alpha_i| = O\left(\frac{\Delta}{\|\boldsymbol{U^i}\|} \right). \tag{7}$$

So, the missing information now is α_i. From Equation (6) and comparing coefficients we obtain a polynomial system of equations I in $\mathbb{Z}[\varepsilon_1, \ldots, \varepsilon_n, \alpha_1, \ldots, \alpha_r]$

$$\begin{aligned} M_1(\varepsilon_1, \ldots, \varepsilon_n) &= F_1 + \textstyle\sum_{i=1}^{r} \alpha_i U_1^i, \\ M_2(\varepsilon_1, \ldots, \varepsilon_n) &= F_2 + \textstyle\sum_{i=1}^{r} \alpha_i U_2^i, \\ &\cdots \\ M_t(\varepsilon_1, \ldots, \varepsilon_n) &= F_t + \textstyle\sum_{i=1}^{r} \alpha_i U_t^i. \end{aligned}$$

Eliminating ε_i from the above ideal I we can derive some new equations

$$H_k(\alpha_i, F_j, U_j^i) \tag{8}$$

depending only on α_i, F_j and U_j^i, $i = 1, \ldots, r$ and $j = 1, \ldots, t$. This process is called in the C++ program `find_relations`; assuming that we have a non linear system of equations, we know that always such that relations do exist and we can compute some of them by substituting the linear monomials into the non linear ones. Of course, we can use Groebner Basis theory or any other elimination method to compute the whole elimination ideal $I \bigcap \mathbb{Z}[\alpha_1, \ldots, \alpha_r]$, but it is quite expensive from the complexity point, see [3]. From Equations (8), we construct a new lattice and apply the same technique as in the first iteration.

This process can be repeated as many times as desired in order to obtain better results.

3.4 Complexity and effectiveness of the algorithm

Let δ be the maximum total degree of the polynomials $f_1, \ldots, f_m \in \mathbb{Z}[x_1, \ldots, x_n]$ and let W be the absolute value of the largest coefficient of $f_i(\Delta_1 x_1, \ldots, \Delta_n x_n)$, $i = 1, \ldots, m$. We are finding algorithms to compute the common zero $(\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{Z}^n$ satisfying the system of polynomial equations (2) in polynomial time in $(\log W, \delta)$, whenever the integers Δ_i are small enough. It is supposed that the given bounds Δ_i depend on $(\log W, \delta)$ and may depend on other parameters. So, the goal in this kind of method is to maximize the bounds up to which roots of the polynomials can be computed in polynomial time. Unfortunately, we are not able to provide an answer of this important aspect. Obviously, if $\max(\Delta_i, i = 1, \ldots, n)$ is small, then the problem become uninteresting.

The *Gaussian heuristic* suggests that an $r-$dimensional lattice with volume $\mathrm{vol}(\mathcal{L})$ is unlikely to have a nonzero vector which is substantially shorter than $\mathrm{vol}(\mathcal{L})^{1/r}$. We analyze the Gaussian heuristic in the first iteration, that is, we want control when $\boldsymbol{E} = \boldsymbol{F}$ and

$$\|\boldsymbol{E} - \boldsymbol{F}\| = O(\Delta) \leq \mathrm{vol}(\mathcal{L})^{1/r}.$$

In order to give a bound for the dimension r of the lattice $\mathcal{L}$, we note that a bound for the cardinality of the set $\mathcal{M}$ is $\binom{n+\delta}{\delta}$. Fixed δ and m then the dimension of the lattice is $O(n^\delta)$ and we have:

$$\Delta \leq \mathrm{vol}(\mathcal{L})^{1/r}.$$

According with the implementation (see Subsection **??**) it is supposed that in the second iteration we obtain a better result, but it seems difficult to study the situation. However, in some concrete polynomials we can provide a deterministic algorithm. We illustrate this fact with the problem of integer factoring, see below Subsection 5.

3.5 The heuristic results in the case of one polynomial

In this section we give an heuristic result for the first and the second iteration when we have a single multivariate polynomial. Suppose that $m = 1$, in that case, we apply our method and we find a vector $\boldsymbol{F}$ with small norm:

$$\|\boldsymbol{F}\| = O(\Delta).$$

If $\boldsymbol{E}$ and $\boldsymbol{F}$ were different, we would have that a lattice analogous to (5) has a relatively small vector:

$$\begin{aligned} a_{11}\mu_1 Y_1 + \cdots + a_{1t}\mu_1 Y_t &= 0, \\ Y_1 &\equiv 0 \mod \tfrac{\Delta}{\mu_1}, \\ &\vdots \\ Y_t &\equiv 0 \mod \tfrac{\Delta}{\mu_t}. \end{aligned} \tag{9}$$

We will denote $\boldsymbol{D} = \boldsymbol{E} - \boldsymbol{F}$. This vector is obviously in the lattice (9). The norm of this vector is:

$$\|\boldsymbol{D}\| = O(\|\boldsymbol{E}\| + \|\boldsymbol{F}\|) = O(\Delta).$$

Now, we are going to bound the volume of the lattice.

Lemma 1. *Let $\mathcal{L}$ be the integer lattice defined by the equations:*

$$\begin{aligned} c_1 Y_1 + \cdots + c_t Y_t &= 0, \\ Y_1 &\equiv 0 \mod \nu_1 \\ &\vdots \\ Y_t &\equiv 0 \mod \nu_t, \end{aligned}$$

where $c_i \in \mathbb{Z}$, $\nu_i \in \mathbb{N}$, $\gcd(c_1, \ldots, c_t) = 1$. Then, we can lowerly bound the volumen of the lattice by:

$$\mathrm{vol}(\mathcal{L}) \geq 2^{-t^2} |c_i| \prod_{j \neq i} \nu_j, \ \forall i = 1, \ldots, t.$$

Proof. If $c_i = 0$, the result is trivial. Otherwise, we define the integer lattice $\mathcal{L}_i$ as the solutions of the system:

$$\begin{aligned} c_1 Y_1 + \cdots + c_{i-1} Y_{i-1} + c_{i+1} Y_{i+1} + \cdots + c_t Y_t &\equiv 0 \mod c_i, \\ Y_j &\equiv 0 \mod \nu_j, \ \forall j \neq i. \end{aligned}$$

Let $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1}\}$ be a basis of $\mathcal{L}$. We build the vectors $U_j \in \mathbb{Z}^{t-1}$, $j = 1, \ldots, t-1$ by just removing the i-th component of $\boldsymbol{u}_j$. It follows easily that $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_{t-1}\}$ generate the lattice $\mathcal{L}_i$. And as c_i does not vanish, it is actually a basis.

Now,

$$\|\boldsymbol{u_1}\| \cdots \|\boldsymbol{u_{t-1}}\| \geq \|\boldsymbol{U_1}\| \cdots \|\boldsymbol{U_{t-1}}\| \geq \mathrm{vol}(\mathcal{L}_i) = |c_i| \prod_{j \neq i} \nu_j.$$

If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1}\}$ is an LLL-reduced basis, then it is known that $\|\boldsymbol{u_1}\| \cdots \|\boldsymbol{u_{t-1}}\| \leq 2^{t^2} \mathrm{vol}\mathcal{L}$. Using this inequality we get:

$$2^{t^2} \mathrm{vol}\mathcal{L} \geq \|\boldsymbol{u_1}\| \cdots \|\boldsymbol{u_{t-1}}\| \geq \|\boldsymbol{U_1}\| \cdots \|\boldsymbol{U_{t-1}}\| \geq \mathrm{vol}(\mathcal{L}_i) = |c_i| \prod_{j \neq i} \nu_j.$$

This finish the proof

Now, another result involving lattice and LLL-reduced basis.

Lemma 2. *Let $\mathcal{L}$ be the integer lattice defined by the equations:*

$$\begin{aligned} c_1 Y_1 + \cdots + c_t Y_t &= 0, \\ Y_1 &\equiv 0 \mod \nu_1 \\ &\vdots \\ Y_t &\equiv 0 \mod \nu_t. \end{aligned}$$

Then, under the Gaussian heuristic, the components of any reduced lattice basis vector is non-zero except the coefficients $(c_1, \ldots, c_{t-1}) \in V \subset \mathbb{Z}_{\nu_t c_t}^{t-1}$ of cardinality:

$$\#V = O\left(\frac{|\nu_1 \nu_t c_t|}{\nu_1 \ldots \nu_t c_t)^{1/(t-1)}}\right).$$

Proof. Let be $\boldsymbol{u_1}, \ldots, \boldsymbol{u_{t-1}}$ a LLL reduced basis. We have seen that if we remove the last component of each vector we have a basis of the following lattice:

$$\begin{aligned} c_1 Y_1 + \cdots + c_{t-1} Y_{t-1} &\mod_{c_t \nu_t} 0, \\ Y_1 &\equiv 0 \mod \nu_1 \\ &\vdots \\ Y_{t-1} &\equiv 0 \mod \nu_{t-1}. \end{aligned}$$

Suppose that there is a vector $\boldsymbol{u_i}$

$$\boldsymbol{u_i} = (\nu_1 e_1, \ldots, \nu_{t-1} e_{t-1})$$

which has one of its components equal to zero. Without loosing generality, we will suppose that the second component is zero and the first one is non zero. Thanks to the Gaussian heuristics, we know that:

$$\|\boldsymbol{u_i}\| = O(\nu_1 \ldots \nu_t c_t)^{1/(t-1)}).$$

It is easy to bound the components of the vector:

$$|e_i| = O\left(\frac{\|\boldsymbol{u_i}\|}{\nu_i}\right).$$

If we fix all the $e_i, c_2, \ldots, c_{t-1}$ there are only $O(\nu_1)$. So the total number possibilities is:

$$\#V = O\left(\frac{|\nu_1 \nu_t c_t|}{\nu_1 \ldots \nu_t c_t)^{1/(t-1)}}\right).$$

This finish the proof.

It is time we applied this theorem to our case. By the Gaussian heuristic and without taking care of constants, we have that if:

$$\|\boldsymbol{E}\|^t = \Delta^t \leq \frac{W\Delta^t}{\prod_{j\neq i}\mu_j} \leq \mathcal{V}(\mathcal{L}_1),$$

then the vector $\boldsymbol{E}$ has many possibilities of being the only one. This inequality can be transform in this one:

$$\prod_{j\neq i}\mu_j < W.$$

Each $\mu_j \leq (\Delta_1 \dots \Delta_n)^\delta$ where δ is the total degree of the polynomial. The number of μ_j is t, which is less than δ^n, so:

$$(\Delta_1 \dots \Delta_n) \leq W^{1/\delta^{n+1}}. \tag{10}$$

or

$$(\Delta_1 \dots \Delta_n) \leq W^{1/t\delta}. \tag{11}$$

For the second iteration we are going to suppose that the polynomial, at least, one monomial of degree two. Let be that monomial $x_r x_s$. After the first step we have:

$$\begin{array}{r} \varepsilon_r = F_a + \sum_{i=1}^t \alpha_i U_a^i, \\ \varepsilon_s = F_b + \sum_{i=1}^t \alpha_i U_b^i, \\ \varepsilon_r\varepsilon_s = F_c + \sum_{i=1}^t \alpha_i U_c^i. \end{array} \tag{12}$$

From this equations it is very clear how to eliminate $\varepsilon_r, \varepsilon_s$, the bounds on the solutions are clear as well, and if we look at the coefficient of the monomial α_t^2 is:

$$1 < U_a^t U_b^t = O\left(\frac{(\|\boldsymbol{U^t}\|\mu_t)^2}{\Delta^2}\right).$$

We are going to apply now again the result of 11 and 7:

$$\frac{\Delta^t}{\mathrm{vol}(\mathcal{L})} = |\alpha_1|\dots|\alpha_t| \leq \left(\frac{\|\boldsymbol{U^t}\|\mu_t}{\Delta}\right)^{2/t(t-1)} (\Delta_{Ut}\Delta_{Ut})^{1/t(t-1)} \approx \mu_t^{2/t(t-1)}.$$

Note that this result is better than the previous one because when $\Delta^t < \mathrm{vol}(\mathcal{L})$ the result trivally holds. In the other case, we are going to use that this approximation $\mu_t \approx (\Delta_1 \dots \Delta_n)^{d/n}$:

$$(\Delta_1 \dots \Delta_n)^{dt - 2\delta n/(t(t-1))} \leq W.$$

or

$$(\Delta_1 \dots \Delta_n) \leq W^{\frac{(t(t-1))n}{(n\delta t^2(t-1)-2d)}}.$$

This is the result for the second iteration.

4 Numerical tests

In this section we summarize some tests on the explained algorithm. We have used ramdomly chosen polynomials whose coefficients (but possibly the independent term) are 100-bits numbers. Once a polynomial and a tolerance Δ are selected, we have performed several test changing the independent term in such a way that the polynomial has a root whose components are bounded by Δ.

As expected, the algorithm finds the corresponding root when the maximum absolute value of the root components Δ is smaller than a certain threshold, which should depend on the number of variables n, the number of appearing monomials $t+1$, and the polynomial degree δ.

According to the algorithm, a maximum number of rounds must be fixed, in such a way that an experiment fails whenever the last round does not output the expected root. In these tests, we only perform two rounds.

We collect in the following tables the empirically obtained threshold expressed as a power of 2^{100}.

$\delta = 2$

$t-n$ \ n	2	3	4
1	0.63	0.36	0.25
2	0.37	0.24	0.18
3	0.28	0.17	0.12
4	$--$	0.15	0.1
5	$--$	0.11	0.08

$\delta = 3$

$t-n$ \ n	2	3	4
1			
2	0.23	0.26	
3	0.22	0.17	
4	0.15	0.14	0.07
5	0.14	0.11	0.03

$\delta = 4$

$t-n$ \ n	2	3	4
1			
2			
3			
4	0.16		
5	0.1	0.08	

5 Integer factoring with extra information

This section is devoted to apply the previous technique to the problem of factoring an integer when we know the high-order bits of one of the factors. We consider a number N which is product of two primes: P and Q and we analyze the assumption that factoring is computationally difficult when the cryptanalyst has access to extra information.

In cryptographic applications, the cryptanalyst may have available additional information above and beyond the number N itself. In practice, Alice or Bob (one of them) typically knows P and Q already, and uses these factors implicitly and/or explicitly during her/his cryptographic computations. The results of these computations may become known to the cryptanalyst, who thereby may find himself at an advantage compared to a pure factoring situation. The necessary information and timing measurements may be obtained by passively eavesdropping on an interactive protocol. The Chinese Remainder Theorem (CRT) is also often used to optimize RSA private key operations. With CRT, $y \bmod P$ and $y \bmod Q$ are computed first (being y is the message to send). These initial modular reduction steps can be vulnerable to timing attacks. The simplest such attack is to choose values of y that are close to P or to Q, and then use timing measurements to determine whether the guessed value is larger or smaller that the actual value of P and Q.

Suppose that an attacker is able to find the high-order h bits of the smallest prime P, t *can we recover P and Q in polynomial time in* $\log N$*?*

A directed application of this problem comes from paper [27]. Here, an identity-based variant of RSA in which the user's modulus N is related to his identity is proposed. For example, the high-order bits of N may be the user's name encoded in ASCII. If N is generated in such a way, somewhat more than the high order $\frac{1}{4}\log_2 N$ bits of P are revealed to the public.

From now, given an integer number A, $\log A$ means $\log_2 A$ and polynomial time means polynomial in $\log N$.

Regarding positive answers for this problem, we can mention the work of Rivest and Shamir [26] using a special case of integer programming, which needs about $h = \frac{1}{3}\log N$ bits of P. The paper [11] by Coppersmith used a lattice-based method to factor N using $h = \frac{3}{10}\log N$ bits of P.

The king result is also due to Coppersmith [9, 10] which requires only $h = \frac{1}{4}\log N$ bits of P and also uses lattice reduction techniques. In fact, his result is

a consequence of Theorem . We do not know any efficient implementation of the algorithm, (see [20] for particular cases), which is quite involved.

5.1 Notation and preliminaries

Given a number N which is product of two primes P and Q, we suppose that $P < Q$. Our results involve a parameter Δ which measures how many high-order bits of P are known. This parameter is assumed to vary independently of P subject to satisfy the inequality $\Delta < P$. More precisely, we say that an integer w is a Δ-*approximation* to integer u if $|w - u| \leq \Delta$.

So, in the case where the high-order h bits of the prime P are given, we can build a Δ-approximation P_0 to P, by taking the h high-order bits of P and $\lfloor \log P \rfloor + 1 - h$ zeroes. In this case, $\Delta = 2^{\lfloor \log P \rfloor + 1 - h} - 1$, that is,

$$\begin{aligned} P - P_0 &\leq \Delta, \\ \Delta &\cong \frac{P}{2^h}. \end{aligned}$$

By dividing N into P_0, we obtain a Δ_1-approximation Q_0 to Q:

$$\begin{aligned} |Q - Q_0| &\leq \Delta_1, \\ \Delta_1 &\cong \frac{Q\Delta}{P}. \end{aligned}$$

Let $\varepsilon_0 = P - P_0$ and $\varepsilon_1 = Q - Q_0$. From $N = PQ$ we obtain:

$$f(\varepsilon_0, \varepsilon_1) = 0,$$

where

$$f(\varepsilon_0, \varepsilon_1) = (P_0 + \varepsilon_0)(Q_0 + \varepsilon_1) - N.$$

And with

$$|\varepsilon_0| \leq \Delta, \quad |\varepsilon_1| \leq \Delta_1. \tag{13}$$

The main objective is to find roots of this innocent polynomial $f(\varepsilon_1, \varepsilon_2)$. Following the strategy in previous section, we obtain the following result:

In order to apply the Theorem 2, we suppose that we know $N = PQ$ and the high-order $h = \frac{1}{4}\log_2 N$ bits of P. We apply Theorem 2 to polynomial $f(\varepsilon_0, \varepsilon_1)$ given by Equation (5.1) and take:

$$\begin{aligned} |x_0| < P_0 N^{-1/4} &= \Delta, \\ |y_0| < Q_0 N^{-1/4} &= \Delta_1, \\ \delta = 1, \quad W &= N^{3/4}. \end{aligned}$$

As corollary it follows:

Theorem 2 (Theorem 4-[9]). *In polynomial time we can find the factorization of $N = PQ$ if we know the high-order $(\frac{1}{4}\log_2 N)$ bits of P*

The proof of this result is quite involved and uses lattice reduction techniques. In one hand, the proof requires the calculation of the associated lattice volume. The corresponding matrix has $(k+\delta)^2$ rows and $(k+\delta)^2 + k^2$ columns, where k is an integer satisfying

$$k > \frac{2}{3\epsilon}.$$

Fixed the degree δ a bound for the dimension of the lattice is $O(k^2)$.

Remark 1. Given $h = (\frac{1}{3}\log_2 N)$ bits of P we want to find the appropriate integer k.

We know that $\Delta\Delta_1 < W^{2/(3\delta)-\epsilon}2^{-14\delta/3}$ and since $W^{3/4}$, we obtain

$$(\frac{1}{4} + \frac{3}{8}\epsilon)\log_2 N < h = \frac{1}{3}\log_2 N.$$

Then $\epsilon < \frac{2}{9}$. On the other hand, $k > \frac{2}{3\epsilon}$ implies $k > 3$. So the lattice dimension is bigger than 41.

In practice, if the number of bits from P is near the threshold $\frac{1}{4}\log N$, the size of intermediate steps grow. This is, less bits known imply more computations. The used lattice dimension is big, as well as the size of lattice coefficients. Finally, his method requires the resultant of two bivariate integer polynomials.

In next subsection we present a new approach where for $\frac{1}{3}\log_2 N$ bits of P we need a lattice of dimension only 4.

5.2 One round lattice reduction

First of all, we remark that if P_0 and Q_0 are given as in Subsection 5.1, they are unique, that is, if we know the high-order h bits of P and N is fixed, then P_0 and Q_0 are uniquely determined. Now we present the main result in this subsection:

Theorem 3. *For a prime P and natural numbers g and h, there is a set $\mathcal{V}(P,g,h) \subseteq \mathbb{Z}_P$ of cardinality $\#(\mathcal{V}(P,g,h)) = O(2^{2\log P+g-3h})$ with the following property. Given $N = PQ$ and the high-order h bits of P, whenever $Q \notin \mathcal{V}(P,g,h)$, there exists an algorithm recovering P and Q on deterministic polynomial time, where $g = \lfloor \log Q \rfloor$.*

Proof. First, we will suppose the number of bits of the prime P is given, then we know g. Let $P_0, Q_0, \Delta_1, \Delta$ as in Subsection 5.1, where P_0 is a Δ-approximation to P and Q_0 is a Δ_1-approximation to Q. We can reformulate the theorem as follows: there is a set $\mathcal{V}(\Delta,\Delta_1) \subset \mathbb{Z}_P$ of cardinality $\#(\mathcal{V}(\Delta,\Delta_1)) = O(\Delta^2\Delta_1)$ with the following property. Whenever $Q \notin \mathcal{V}(\Delta,\Delta_1)$, there exists an algorithm recovering P and Q on deterministic polynomial time.

The result is trivial when $4\Delta^2\Delta_1 \geq P$, and so we assume $4\Delta^2\Delta_1 < P$.

The set $\mathcal{V}(\Delta, \Delta_1)$ of primes Q that we are going to exclude consists of values Q satisfying the following congruence:

$$d_1 Q + E \equiv 0 \mod P, \tag{14}$$

where $|d_1| \leq 4\Delta$ and $|E| \leq 8\Delta\Delta_1$. Note that there are at most $O(\Delta^2\Delta_1)$ choices for d_1 and E. Once these parameters are chosen, there can be at most one choice for Q such that $d_1 Q + E \equiv 0 \mod P$. Hence, $\#(\mathcal{V}(\Delta, \Delta_1)) = O(\Delta^2\Delta_1)$.

Suppose that $Q \notin \mathcal{V}(\Delta, \Delta_1)$. An outline of our proof goes as follows. We aim to show that the integers ε_j occur as certain components of a short vector in an appropriate lattice; this lattice can be constructed from the information that we are given. We find ε_0 and ε_1 by using well-known techniques for finding short vectors in lattices, and then we use the equalities $P = \varepsilon_0 + P_0$ and $Q = \varepsilon_1 + Q_0$ to recover P and Q. We obtain

$$f(\varepsilon_0, \varepsilon_1) = 0.$$

More explicitly, the method is derived from the following construction.

$$\begin{aligned} A = P_0 Q_0 - N,\ B = Q_0 \Delta, \\ C = P_0 \Delta_1,\ D = \Delta\Delta_1, \end{aligned}$$

then

$$A\Delta\Delta_1 + B\Delta_1\varepsilon_0 + C\Delta\varepsilon_1 + D\varepsilon_0\varepsilon_1 = 0. \tag{15}$$

Therefore, the lattice $\mathcal{L}$ consisting of integer solutions $\boldsymbol{x} = (x_0, x_1, x_2, x_3) \in \mathbb{Z}^4$ of the system of congruence equations:

$$\begin{aligned} &Ax_0 + Bx_1 + Cx_2 + Dx_3 = 0, \\ &x_0 \equiv 0 \mod \Delta\Delta_1, \\ &x_1 \equiv 0 \mod \Delta_1, \\ &x_2 \equiv 0 \mod \Delta, \end{aligned} \tag{16}$$

contains the vector

$$\boldsymbol{e} = (\Delta\Delta_1 e_0, \Delta_1 e_1, \Delta e_2, e_3) = (\Delta\Delta_1, \Delta_1\varepsilon_0, \Delta\varepsilon_1, \varepsilon_0\varepsilon_1). \tag{17}$$

We aim to show that $\boldsymbol{e}$ is a small vector in the lattice $\mathcal{L}$. We have:

$$e_0 = 1, \qquad |e_1| \leq \Delta \qquad |e_2| \leq \Delta_1, \qquad |e_3| \leq \Delta\Delta_1.$$

Using the bounds given in Equation (13), the Euclidean norm of $\boldsymbol{e}$ satisfies the inequality

$$\|\boldsymbol{e}\| \leq \sqrt{\Delta^2\Delta_1^2 + \Delta^2\Delta_1^2 + \Delta^2\Delta_1^2 + \Delta^2\Delta_1^2} = 2\Delta\Delta_1. \tag{18}$$

Assume there is another vector $\boldsymbol{f} = (\Delta\Delta_1 f_0, \Delta_1 f_1, \Delta f_2, f_3) \in \mathcal{L}$ with

$$\|\boldsymbol{f}\| \leq \|\boldsymbol{e}\| \leq 2\Delta\Delta_1.$$

which is not parallel to $\boldsymbol{e}$, in particular:

$$|f_0| \leq 2, \qquad |f_1| \leq 2\Delta \qquad |f_2| \leq 2\Delta_1, \qquad |f_3| \leq 2\Delta\Delta_1. \tag{19}$$

We define the vector $\boldsymbol{d} = f_0\boldsymbol{e} - e_0\boldsymbol{f}$. The first component of the vector $\boldsymbol{d}$ is zero, and $\boldsymbol{d}$ lies in the lattice $\mathcal{L}$. Then, the first congruence in Equation (16) implies that

$$B\Delta_1 d_1 + C\Delta d_2 + Dd_3 = 0.$$

Simplifying the above equation:

$$Q_0 d_1 + P_0 d_2 + d_3 = 0, \tag{20}$$

where $d_i = e_i f_0 - f_i$. Note that $|d_i| \leq 2|e_i| + |f_i|$ for $i = 1, 2, 3$ and so our bounds on e_i and f_i imply

$$|d_1| < 4\Delta, \quad |d_2| < 4\Delta_1, \quad |d_3| < 4\Delta\Delta_1. \tag{21}$$

From Equation (20), if $d_1 = d_2 = 0$ then $d_3 = 0$. But this implies $\boldsymbol{d} = \boldsymbol{0}$ and so $f_0\boldsymbol{e} = e_0\boldsymbol{f}$. This contradicts the fact that $\boldsymbol{f}$ and $\boldsymbol{e}$ are not parallel.

The case $d_1 = 0 \neq d_2$ is easily avoided because the assumption $4\Delta^2\Delta_1 < p$ implies $p - \Delta > 4\Delta\Delta_1$, and so, the resulting equation $P_0 d_2 + d_3 = 0$ cannot be satisfied.

Making the substitutions $P_0 = P - \varepsilon_0$ and $Q_0 = Q - \varepsilon_1$ in Equation (20) and reducing modulo P, we find that

$$Qd_1 + d_3 - \varepsilon_0 d_2 - \varepsilon_1 d_1 \equiv 0 \mod P. \tag{22}$$

Writing:

$$E = d_3 - \varepsilon_0 d_2 - \varepsilon_1 d_1,$$

we obtain:

$$Qd_1 + E \equiv 0 \bmod P. \tag{23}$$

The bounds (21) imply $|d_1| \leq 4\Delta$ and $|E| \leq 8\Delta\Delta_1$. But then (23) implies that $Q \in \mathcal{V}(\Delta, \Delta_1)$, and so we have a contradiction. This contradiction shows that there exists no small vector $\boldsymbol{f}$ in $\mathcal{L}$ other than vectors parallel to $\boldsymbol{e}$.

We remark that $\mathcal{L}$ is defined using information we are given, and recall that the shortest vector problem can be solved in deterministic polynomial (in the bit size of a given basis of the lattice) time in any fixed dimension, see [19]. This certainly applies to the lattice $\mathcal{L}$. Once we have found a short vector $\boldsymbol{f}$ in $\mathcal{L}$, we know that $\boldsymbol{e} = \boldsymbol{f}/f_0$ since $\boldsymbol{f}$ is parallel to $\boldsymbol{e}$ and since $e_0 = 1$. Obviously, given the second component $\Delta_1\varepsilon_0$ of $\boldsymbol{e}$ we can find P.

To finish the proof, remember that we have assumed to know the number $\lfloor \log P \rfloor$. If we do not have this information we can apply the above algorithm from 1 to $\lfloor \log N \rfloor$. Obviously the time complexity keep polynomial on $\log N$. This completes the proof. □

For practical application of this bound, we note that in most of the cases the values P and Q are taken randomly, then the probability that Q lies in $\mathcal{V}(\Delta, \Delta_1)$ is:

$$\frac{\Delta^2 \Delta_1}{P} = \frac{\Delta^3 Q}{P^2}. \tag{24}$$

And this is less than one if:

$$\Delta^3 < \frac{P^2}{Q}. \tag{25}$$

In other words:

$$\frac{\log N}{3} = \frac{\log P + \log Q}{3} \leq h. \tag{26}$$

If this inequality holds, then we can probably find the complete factorization by this method. In fact this was the first bound obtained by Rivest and Shamir [26]. However, the present algorithm compare favorably computationally speaking with Coppersmith method, because we are working in a lattice of dimension 4 instead of 41, see Remark 1.

5.3 Two rounds lattice reduction

The previous theorem can be generalized to two rounds lattice reductions obtaining a better result following the Algorithm in Subsection 3.2, that is, use the Closest Vector Problem CVP (see Section 2) instead of SVP used in the Theorem 3.

Theorem 4. *For a prime P and natural numbers g and h, there is a set $\mathcal{V}(P, g, h) \subset \mathbb{Z}_P$ of cardinality $\#(\mathcal{V}(P, g, h)) = O\left(2^{4\log P - 12h + 4g}\right)$ with the following property. Given $N = PQ$ and the high-order h bits of P, whenever $Q \notin \mathcal{V}(P, g, h)$, there exists an algorithm recovering P and Q on deterministic polynomial time, where $g = \lfloor \log Q \rfloor$.*

We remark that the size of the "possibly failing" values, when express in terms of Δ, Δ_1 translates into: $\#(\mathcal{V}(P, g, h)) = O\left(\left(\frac{\Delta^2 \Delta_1}{P}\right)^4\right)$.

Now, lets talk about the practical applications of this result, as Theorem 3, we want to know when it probably we can recover this two factors in polynomial time. The result is non trivial if

$$(\Delta^2 \Delta_1)^4 \leq P^5$$

or

$$(\Delta^{12}) \leq \frac{P^9}{Q^4}.$$

In terms of bits, since $\Delta \cong \frac{P}{2^h}$, we have

$$\frac{\log(P)}{4} + \frac{\log(Q)}{3} \leq h.$$

Or, in terms of the number of bits of N:

$$\frac{\log(N)}{4} + \frac{\log(Q)}{12} \leq h.$$

If the primes P and Q are balanced, that is, $P = O(N^{1/2}) = Q$ (for instance due to security reasons this is the case for RSA cryptosystem) we have

$$7\frac{\log(N)}{24} \leq h.$$

This compares favorably with the first result of Coppersmith [11] which requires to know $3\frac{\log(N)}{10}$ bits of P. Lets make an example suppose that P and Q has 500 bits, from [11] result we need to know 300 bits, in our case we need 292 bits. However, our result is worst if the primes are unbalanced.

Remark 2. On the other hand, the dimension of the lattice in [9, 10] for this particular number of bits $7\frac{\log(N)}{24} = h$ is higher than our 6 lattice dimension on Theorem 4. Applying the same argument as Remark 1 we have that

$$(\frac{1}{4} + \frac{3}{8}\epsilon)\log_2 N < h = \frac{7}{24}\log_2 N.$$

Then $\epsilon < \frac{2}{27}$. Since $k > \frac{2}{3\epsilon}$, it implies $k > 9$. So the lattice dimension in $[9, 10]$ is bigger than 199.

We are working in the generalization of our method, we have some problems to compute the volume of the corresponding lattices in order to obtain similar results like previous one. The following subssection will show that when the numbers of rounds grows we obtain a better result, but never we get Coppersmith's bound [9].

5.4 Experimental results

We want to provide some experimental results. We have implemented the algorithm introduced in the previous section on a Debian-PC computer with two processors of 1Ghz and 1Gb (respectively) of memory ram. Our implementation use the free library **NTL** (=Number Theory Library)[25] together with the library **GMP** (=Gnu Multi Precision Library).

In the implementation we have asked ourselves what is the time of finding the factorization of these numbers, we liked to see how our algorithm does in a general case, that is, using the same idea in a general way, and to see if it is practical in its use.
We measure the time the number of bits of P and the number of bits of Q and in number of known P's bits.

Bits of P	Bits of Q	Bits known	Time
100	100	66	3.675 sec
100	100	60	10.271 sec
512	512	306	11.392 sec
512	512	300	14.025 sec
512	512	292	15.339 sec
1024	1024	624	7.240 sec
1024	1024	600	29. 357 sec
1024	1024	580	1 m. 35 sec

All the cases that we have choosen, the algorithm has returned the factorization, the first two cases where case for toying, the next three cases are not trivial, that means that a number of 1024 bits is a big number, and finding its factorization is not easy at all. On the negative side, the results are quite distant from Coppersmith result, the reason of this is that we need to apply more times, however in that case, the constast takes places so, like in Coppersmith method, it become necessary to guess more bits in order to make the algorithm works.

References

1. M. Ajtai, R. Kumar and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem", *Proc. 33rd ACM Symp. on Theory of Comput. (STOC 2001)*, Association for Computing Machinery, 2001, 601–610.
2. L. Babai, "On Lovasz Lattice Reduction and the Nearest Lattice Point Problem", *Combinatorica*, **6**, 1986, 1–6.
3. T. Becker, V. Weispfenning, "Groebner bases. A computational approach to commutative algebra". In cooperation with Heinz Kredel. Graduate Texts in Mathematics, **141**. Springer-Verlag, New York, 1993.
4. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, "Predicting nonlinear pseudorandom number generators", *Math. Computation*, **74** (2005), 1471–1494.
5. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, "Predicting the inversive generator", *Proc. Coding and Cryptography, IMA-03*, LNCS **2898**, Springer-Verlag, Berlin 2003, 264–275.
6. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, "Reconstructing noisy polynomial evaluation in residue rings", *Journal of Algorithms*. In press, S0196-6774(04)- 00115-4/FLA AID:1388. Available online.
7. J. Bloemer, A. May, "A tool kit for Finding small roots of Bivariate Polynomial over the Integers", *Advances in Cryptology-Crypto 2003*, LNCS **2729**, Springer Verlag, 2003, 27–43.
8. J.W.S. Cassels, "An Introduction to the Geometry of Numbers". Springer-Verlag, New York, 1971.
9. D. Coppersmith: "Small solutions to polynomial equations and low exponent RSA vulnerabilities". *J. Cryptology* **10 (4)**, 1997, 233–260.
10. D. Coppersmith: "Finding a Small Root of a Bivariate Integer Equations; Factoring with High Bits Known". U. Maurer (Ed), *Proc. EUROCRYPT-96*, LNCS **1070**, Springer-Verlag, Berlin 1996, 155–156.

11. D. Coppersmith: "Factoring with a hint". *IBM Research Report RC. 1995*, January 16, 1995.
12. J-S Coron, "Finding small roots of Bivariate Integer Polynomial Equations Revisted", *Proc. Advances in Cryptology- Eurocrypt'04*, LNCS **3027**, Springer Verlag, 2004, 492–505.
13. D. Gomez-Perez, J. Gutierrez and A. Ibeas, "Cryptanalysis of the Quadratic generator", *Proceedings in Cryptology-INDOCRYPT 2005*, LNCS **3797**, Springer Verlag, Berlin 2005, 118–129.
14. D. Gomez-Perez, J. Gutierrez and A. Ibeas, "Efficient Factoring Based on Extra Information", Preprint, University of Cantabria, Spain 2006. A preliminary version in Proc. Spanish Conference on Cryptography, RECSI-2006, Barcelona, September 2006.
15. J.W.S. Gruber and C.G. Lekkerkerker, "Geometry of Numbers". North-Holland, 1987.
16. A. Joux and J. Stern, "Lattice reduction: A toolbox for the cryptanalyst", *J. Cryptology*, **11** (1998), 161–185.
17. N.A. Howgrave-Grahm, "Finding small roots of univariate revisted", *Proc. Cryptography and Coding*, LNCS **1355**, Springer Verlag, 1997, 131–142.
18. E. Jochemz and A. May, "A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants" *n Advances in Cryptology (Asiacrypt 2006)*, Lecture Notes in Computer Science, Springer-Verlag, 2006.
19. R. Kannan, "Minkowski's convex body theorem and integer programming", *Math. Oper. Res.*, **12** (1987), 415–440.
20. A. May, "Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring", *In Advances in Cryptology (Crypto 2004)*, LNCS **3152**, Springer Verlag, 2004, 213–219.
21. Paul C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". *Proc. CRYPTO-96*, LNCS **1109**, Springer-Verlag, Berlin 1996, 104–113.
22. A. K. Lenstra, H. W. Lenstra and L. Lovász, "Factoring polynomials with rational coefficients", *Mathematische Annalen*, **261** (1982), 515–534.
23. D. Micciancio and S. Goldwasser, "Complexity of lattice problems", Kluwer Acad. Publ., 2002.
24. P.Q. Nguyen and J. Stern, "Lattice reduction in cryptology: An update", in: W. Bosma (Ed), *Proc. ANTS-IV*, LNCS **1838**, Springer-Verlag, Berlin 2000, 85–112.
25. V. Shoup, "Number theory **C++** library (NTL)", version 5.3.1, available at `http://www.shoup.net/ntl/`.
26. R. L. Rivest and A. Shamir: "Efficient factoring based on partial information". *Advances in Cryptology, Proc. EUROCRYPT-85*, LNCS **219**, Springer-Verlag, Berlin 1986, 31–34.
27. S.A. Vanstone and R. J. Zuccherato: "Short RSA Keys and their Generation". *J. Cryptology* **8 (2)**, 1995, 101–114.